EVALUATING GRAPH THEORETIC CLUSTERING ALGORITHMS FOR RELIABLE MULTICASTING

Esther Jennings †
Jet Propulsion Laboratory
4800 Oak Grove Drive,
Pasadena, California,91109, USA.

Lenka Motyčková, David Carr
Computer Science Department,
Luleå University of Technology,
Luleå, SWEDEN.

Abstract—**In reliable multicast protocols, each data packet being sent must be acknowledged. Collecting the acknowledgments centrally at the sources can cause ACK-implosion and can result in poor scalability. To overcome this, clustering algorithms which use virtual structures to gather acknowledgments were proposed. In this work, we analyze the complexities of three such clustering algorithms: Lorax, k-degree, and Self-adjust. We compare the quality of the virtual structures produced by these algorithms, focusing on the number of clusters, cluster size, cluster radius, and the optimal positioning of cluster leaders. Our simulation showed that the virtual structure produced by Self-adjust is better in terms of cluster radius and the location of cluster leaders. However, due to the self-adjusting nature of the algorithm, it might take longer time to compute than the other two algorithms.**

I. Introduction

With the growing size of the Internet, as well as the growing interest in multicast applications, we can expect large multicast groups and groups that span a large geographical area. The routers in the Internet are responsible for routing multicast packets. Thus, the growth of multicast groups usually increases the load of all the intermediate routers and may cause an overload on routers. To avoid overload, multicast protocols must be scalable, i.e., the network load must be independent of the number of receivers in a multicast group. In reliable group multicast, each packet being sent must be acknowledged to ensure that the packet has been received by all the multicast group members. Collecting the acknowledgments centrally at the sources can cause ACK-implosion and can result in poor scalability. To overcome this, several clustering algorithms were proposed which use virtual structures to gather acknowledgments. These virtual structures might contain interconnected disjoint clusters, or a multi-level hierarchy of overlapping clusters.

Clustering algorithms have been studied extensively since the 1970's [1], [2]. With the recent interest in data mining and computational biology, there is a new wave of clustering algorithms being proposed [3], [4], [5], [6]. Some of these algorithms use graph theoretical techniques to partition an input graph into disjoint clusters. This is very similar to the clustering technique being used in distributed computing to achieve concurrency and scalability.

In distributed computing, a clustering algorithm would partition the graph into local groups according to the *local connectivity properties* of the graph. Applying this technique recursively, we can obtain a hierarchical clustering where a higher-level cluster can contain several lower-level clusters. Using this hierarchy, one can minimize the amount of communication by performing computations as much as possible locally and sending only aggregated data globally, where the aggregated data is a summary representing all the data in a cluster.

Since reliable multicast services are being used increasingly to support real-time applications, it becomes necessary to understand the strengths and weaknesses of the virtual structures produced by the previously proposed clustering algorithms. The quality of these structures may directly affect the performance of reliable multicast protocols which use these structures to gather acknowledgments.

In recent years, several clustering algorithms have been proposed [7], [8], [9], [10], [11]. The algorithms proposed in [7], [10], [11] produce a clustering, which reflects the local connectivity properties of the network (or multicast group) topology. The algorithms proposed by [8], [9] produce a clustering which does not necessarily preserve the local connectivity properties. That is, a real cluster (densely connected subgraph) can be cut into several pieces, where each piece ends up in a different cluster.

In these previous works, performance was evaluated in terms of network throughput only. The virtual structures were integrated into reliable multicast protocols and performance was evaluated by analysis and simulation. In [7], [8], [9], [10], [11], the throughput was analyzed in terms of the load per node, which is bounded by a constant. Thus, the throughput alone does not tell us how the virtual structures produced by the different clustering algorithms compare against each other. Without looking at the properties of the virtual structures, we gain no insight about how these properties may affect the performance of reliable multicast protocols.

This work compares, by simulation, the virtual structures produced by three selected clustering algorithms: *Lorax*, *k-*

† The research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

degree and *Self-adjust* clustering. To evaluate the quality of the virtual structures, We focus on the following metric: (1) number of clusters: affects the load at the source(s); (2) cluster size: affects the load of the cluster leader; (3) cluster radius: affects the delay within the cluster; and (4) optimality: accesses the placement of a leader in relation to its cluster members with respect to a tree structure. The leader position can have an effect on the retransmission delay. If an acknowledgment-tree (Ack-tree) is used for gathering acknowledgments, the Ack-tree achieves best performance if the cluster leaders are also the ancestors of their respective cluster nodes in the multicast delivery tree. In this measure, we count the number of cluster members which are not descendents of the cluster leader in the multicast tree.

II. Multicast Group Modeling

We model the network as a graph containing routers only, because the receivers are directly connected to the routers as peripheral trees. We assume the existence of multicast tree(s) for the routing of multicast data. The cluster structure produced by the clustering algorithms are used to collect acknowledgments for the multicast packets.

Given an underlying shared multicast tree, graph G is defined as the (connected) subgraph of the network, induced by the vertices of the underlying multicast routing tree(s). Note that, G contains all the vertices of the multicast routing tree(s) plus all other IP-network connections induced by these vertices.

III. Analysis of Algorithms

We focus on graph theoretic clustering algorithms which are suited for distributed computation using only local information. For example, graph theoretic clustering algorithms which compute minimum cuts are not suitable for distributed computing because two edges belonging to the same cut may be very far away from each other, and they might not share any common end-points. Thus, one cannot compute the cuts from local information only.

In RMTP receivers are grouped into local regions. There is a designated receiver (DR) in each region to assist the sender to process acknowledgments and to retransmit data. However, we did not simulate the RMTP clustering because we do not know how to implement the selection of DRs in order to make a fair comparison.

In [7], a greedy algorithm which forms clusters of cliques (fully connected graphs) was presented. Packets are routed over shortest paths between boundary nodes in cliques. We did not simulate this algorithm because the Internet tends to be sparse, so the clique size is usually small. Therefore, this type of clustering does not help in decreasing the load of the routers by much.

Before we present the three selected algorithms, a few terminologies are need. The time complexity is estimated by assuming that each message takes $O(1)$ time to transmit; this does not imply that the algorithms are synchronous. The messages are short because each message contains a constant number of parameters. The message complexity estimates the number of messages exchanged in the algorithms for clustering. Table I summarizes the complexities of the simulated algorithms, where m and n are the number of edges and the number of nodes of an input graph G respectively. The Ω in the table denotes the lower bound on time or messages with respect to each algorithm. The analysis of the algorithms are presented in Section III.

TABLE I
Algorithm Complexities

Complexity	Lorax	k-degree	Self-adjust
Time	$\Omega(\log_B n)$	$\Omega(k \cdot D_{max})$	$\Omega(\log_k n)$
Message	$\Omega(m)$	$\Omega(m)$	$\Omega(m)$

A. Lorax Algorithm

This algorithm [8] first constructs a shared Ack-tree starting from a specific root node. The Ack-tree is constructed using both root-based and off-tree based schemes. The algorithm is root-based because the Ack-tree is grown from a specific-selected root using the expanded ring search (ERS) heuristic. The ERS is controlled by a time-to-live (TTL) parameter which is a hop-count. That is, the ERS will only reach nodes within TTL hops. The root-based ERS alone does not guarantee that all the receivers wanting to receive reliable multicast packets will be reached. This can happen if many nodes between the root and the receivers are uninterested in getting reliable multicast packets. Therefore, we need the off-tree based scheme where a receiver can initiate an ERS to find nodes on the Ack-tree, and to attach itself to the Ack-tree through one of these nodes.

Clusters are formed by grouping a node with at most B of its children, where B is a parameter. A child is any node which can be reached within a pre-specified delay. Each node may have a different parameter B. The node and its (at most B) children form a local group. If a node has more than B children, it will abandon the extra children. The process of keeping the number of children at or below B at each node is called *fission*. The abandoned children become orphans. If the orphans were to look for new parents using the ERS heuristics, this may cause an non-scalable amount of work for some nodes. Therefore, the node which abandons children will look among its remaining children to find the node (say v) which has the least number of children. The node will ask v to initiate an adoption request. Hopefully, this node is also close to the orphans. The adoption request is sent to v's local group, hoping to get responses from the orphans. By this fission process, the clusters formed might not be topology preserving. That is, a node might need to pass another cluster to reach other nodes in the same cluster. This affects the retransmission; if the cluster leader is not an ancestor of the node requesting re-

transmission, the request might be propagated to the root of the ACK-tree, making retransmissions non-local.
Time complexity : assume the input graph is a tree where each node has B children within distance one. The virtual structure will then have $log_B\ n$ layers, thus the time complexity of $\Omega(\log_B\ n)$.
Message complexity : assume a predefined root starts the algorithm. The clusters are formed during an ERS to reach nodes at the next layer. Since we assume each node has B children at distance one, a node starting to grow a cluster will finish within constant time. The number of messages sent on each link to form clusters is bounded by a constant. Thus, the message complexity is $\Omega(m)$.

B. k-degree Algorithm

The algorithm is given two parameters, k, and D_{max}, where k is a guess of the highest node degree, and D_{max} bounds the cluster radius. The algorithm detects nodes of degree at least k (having at least k neighbors) and builds a cluster around each such node. The clusters are built concurrently, so they may compete for members. Starting from a node (of degree $\geq k$), nodes are added to the cluster layer by layer. Let the number of nodes in the current layer be $|layer_{current}|$. If there are $k \times |layer_{current}|$ available nodes in the next layer, and the cluster radius has not reached D_{max} yet, then the cluster continues its growth by including the available nodes in the next layer. Otherwise, it stops at the current layer. The algorithm then decreases k by one, and repeats the cluster building process, until $k = 3$. After this, the remaining nodes not belonging to any clusters must belong to a chain. Chains are then divided among their closest clusters.
Time complexity : assume all nodes have degree k. Each cluster takes $\Omega(D_{max})$ to create, and the clusters are formed concurrently. The process is repeated for degree $(k-1)$, $(k-2)$, $\cdots$, 3. This totals to $\Omega((k-3)\cdot D_{max})$ time. No synchronization is needed between passes because a timeout can be used to start the next pass. If D_{max} is $\log_k\ n$, and all nodes have degree k, then the time complexity would be $\Omega(\log_k\ n)$.
Message complexity : the algorithm makes $(k-3)$ passes to grow clusters around nodes of degree $k, (k-1), (k-2), \cdots, 3$. In each pass, the nodes not belonging to a cluster yet which satisfies the degree requirement will start growing its cluster. The nodes already belonging to clusters do not exchange messages. When forming a cluster, the nodes involved exchange a constant number of messages on each edge. Thus, the message complexity is $\Omega(m)$.

C. Self-adjust Algorithm

Assume a tree is given; this tree can be a multicast tree or the ACK-tree of [8]. Starting from the root, the root becomes active, and becomes the root of the first cluster. Then the active nodes advertise within a predefined distance, D_{act}, to ask other nodes to join its cluster. If it does not hear any reply from any node wanting to join its cluster, then it will advertise again; this time *doubling* the distance D_{act} (as long as D_{act} is less than the maximum distance D_{max}). D_{max} is a guess on the maximum radius of the given tree. A parameter, K_{max}, is used to limit the number of replies (similar to B in the *Lorax* algorithm).

If an advertising node receives too many replies (that is, exceeding K_{max}) from distinct nodes wanting to join its cluster, then it checks whether D_{act} can be *halved*. If so, the node will reject all the nodes wanting to join its cluster, decrease D_{act} to half of the old value, and advertise again. However, when the number of replying nodes is acceptable, the new leader starts the second level clusters within the radius where too many nodes first replied.

If D_{act} is one and a node receives more than K_{max} replies, then it is assumed that the node is capable of handling all replying nodes. So, it accepts all nodes.

If a node has replied to a potential leader and it is waiting for a reply, or if it is advertising to be a leader, then it does not forward advertisements from other potential cluster leaders. In this way, nodes on the edge of each cluster will block interior nodes from becoming leaders.
Time complexity : assume all nodes have degree k, then the time complexity would be $\Omega(\log_k\ n)$.
Message complexity : since we assume each node has degree k, the algorithm will create clusters successfully by exchanging a constant number of messages on each edge. Thus, the message complexity is $\Omega(m)$.

IV. Simulation

For our simulation, we generate input graphs that have Internet-like topologies by using Tiers (version 1.2)[12]. The same graphs are used for all three algorithms. We run the simulation using Java version 1.2 on a Sun Sparc station. Each topology is a graph G representing a multicast group. We could regard G as the connected subgraph of a network, induced by the vertices of predefined multicast routing tree(s).

The goal of a good clustering algorithm is to obtain a balanced clustering structure (dependent on the topology of the multicast groups in the network). The structure should enable different clusters to process acknowledgments concurrently, and to localize retransmissions. That is, when a packet is missed at a node, the lost packet will be obtained from another node which resides in the same cluster or at a nearby cluster whenever possible. The simulation assumes: (a) all links have equal delays, (b) no messages are lost, and (c) all nodes have equal response times.

We generate topologies containing 5 MANs (60 and 110 routers), 10 MANs (115 and 215 routers), 20 MANs (225 and 425 routers), 50 MANs (570 and 1070 routers), 100 MANs (1350 and 3150 routers), 200 MANs (2700 routers). Table II summarizes the parameters used to generate the input graphs. In the table, nR is the number of router nodes, $nMAN$ is the number of MANs, $nLAN$ is the number of LANs per MAN, $wBone$ is the number of back-bone nodes connecting the MANs to the WAN, and $mBone$ is the number of nodes connecting the LANs to MANs. The redundancy parameter in

the WAN is set to three for all input graphs because we want to generate graphs that are more connected than trees. All other redundancy parameters are set to one.

TABLE II
TOPOLOGY GENERATION PARAMETERS

Graph	nR	nMAN	nLAN	wBone	mBone
G_1	60	5	5	10	5
G_2	110	5	10	10	10
G_3	115	10	5	15	5
G_4	215	10	10	15	10
G_5	225	20	5	25	5
G_6	425	20	10	25	10
G_7	570	50	5	70	5
G_8	1070	50	10	70	10
G_9	1350	100	5	150	7
G_{10}	3150	100	10	150	20
G_{11}	2700	200	5	300	7

When running the simulation, the following parameters are used for all the runs: $D_{init} = 4$ (this is the initial value of D_{act}), $D_{max} = 10$ (a guess of the maximum radius of the tree, and by the d-degree algorithm to bound the radius of a cluster), B_{max} = 6 (the maximum number of children within a fixed distance, used by *Lorax*), $dLim = 6$ (used as k in the k-degree algorithm). When simulating *Lorax* and the *Self-adjust* algorithms, if the graph has n nodes, we make n runs. In each run, a different node is used as the root to start the algorithm.

V. RESULTS

From the simulation, we observe the following:

- **cluster number**: *k-degree* produced least number of clusters; *Self-adjust* produced the most number of clusters. Fig. 1 shows the number of clusters generated by the algorithms. On the horizontal axis are the input graphs G_1 to G_{11}, where G_1 has the smallest number of MANs and G_{11} the largest number of MANs. The vertical axis is used for the average number of clusters over all simulation runs. In all three algorithms, the number of clusters increases as the number of MANs increases. This is an indication that the virtual structures produced by the algorithms cannot be readily used to achieve scalable reliable multicast protocols. In a scalable protocol, the number of clusters needs to be kept at a small constant. This means, we may need to relax the other parameters, e.g., cluster radius, cluster size.
- **cluster size**: since both *Lorax* and *Self-adjust* have control over the maximum cluster size, the cluster sizes for these algorithms are kept at a constant 7. On the other hand, the cluster size produced by *k-degree* fluctuates between 10 and 20. Looking at the result of *k-degree*, there is no clear trend of increased cluster size as the number of MANs increases. Instead, the clusters formed depend highly on the node degree (topology-dependent). For input graphs with the same number of MANs, the graphs with a higher number of LANs per MAN produce clusters of larger size, which is what we expected.

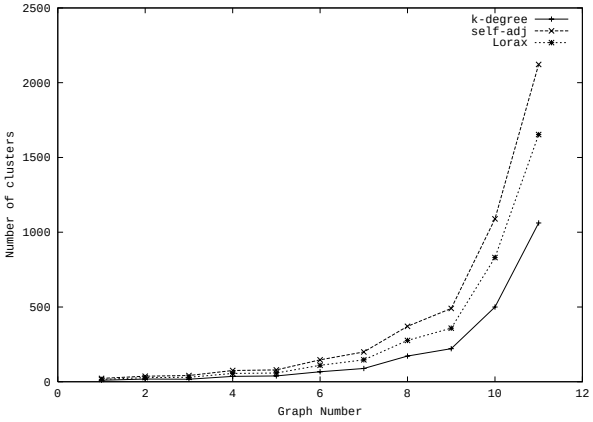

Fig. 1. The number of clusters with respect to graphs G_1 to G_{11}.

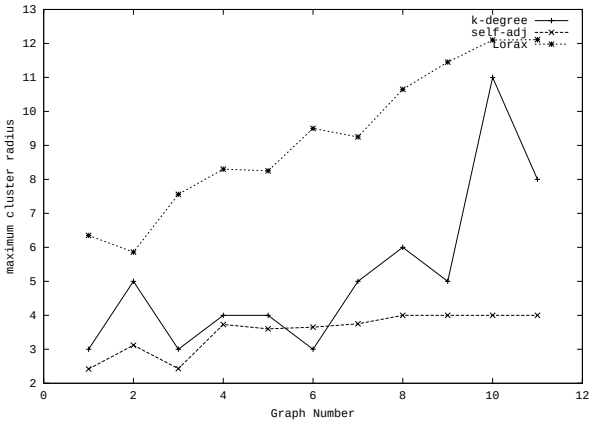

Fig. 2. Maximum cluster radius with respect to graphs G_1 to G_{11}.

- **cluster radius**: In Fig. 2, we compare the maximum cluster radius produced by the algorithms. On the horizontal axis are the input graphs G_1 to G_{11}, where G_1 has the smallest number of MANs and G_{11} the largest number of MANs. The vertical axis is used to record cluster radius. These are the average of maximum cluster radii over all runs. Using *Lorax*, the cluster radius increases as the number of MANs increases. Using *k-degree*, the cluster radius fluctuates because the clusters produced are highly degree-dependent (topology-dependent). For the *Self-adjust* algorithm, the cluster radius increases slightly as the number of MANs increases. When the number of MANs reached a certain number (approximately. 50), the cluster radius becomes constant.
- **optimality**: *Self-adjust* strives to preserve topology, and under the ideal conditions of the simulation, all cluster leaders were ancestors of their respective members. The *k-degree* algorithm does not even assume an underlying multicast routing tree, so optimality has no meaning here. For *Lorax*, Fig. 3 shows that the number of non-optimal nodes increases more than linearly as the number of MANs increases. This could affect the reliable multicast protocol unfavorably in terms of

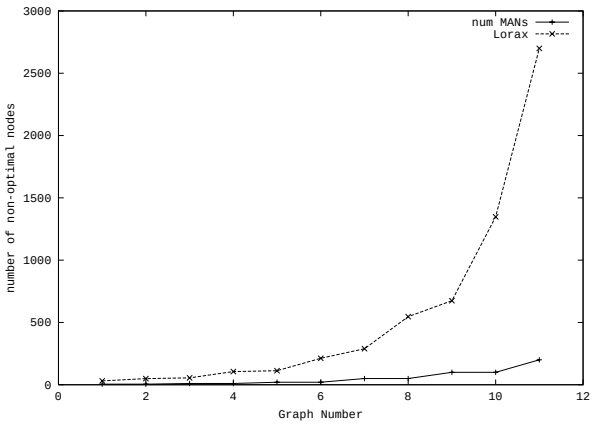

Fig. 3. The number of non-optimal nodes with respect to graphs G_1 to G_{11}.

transmission delays and the locality of retransmissions.

The simulation result is produced from 11 runs of *k-degree* (one per graph), 9,990 runs of *Lorax* (one per node, as root, for each graph) and 9,990 runs of *Self-adjust*.

VI. Conclusion

From the simulation runs, we observed a clear trade-off between the number of clusters and cluster size, as we expected. To achieve optimality, we motivate the use of topology preserving clustering algorithms. The *k-degree* algorithm is good for producing a smaller number of clusters, but the cluster size is larger. *Self-adjust* is good for producing small-sized clusters, but it produces a larger number of clusters. From the stand-point of scalability, we suggest that future clustering algorithms should strike a good balance between cluster number and cluster size, and use a multi-level hierarchy so that acknowledgments can be gathered within each cluster and level by level. In our simulation, we had only tried one set of parameters because we want to evaluate the cluster structures produced by the different algorithms under similar constraints. We need to simulate the algorithms more extensively using other parameter values.

As future work, we propose to prove the optimality of the *Self-adjust* algorithm, and to study the optimization of different parameters while considering network protocol design criteria. The criteria might vary depending on the network applications. In our simulation, we assumed no loss and the same delay on all the links. In the future, we need to consider more realistic assumptions, dynamically changing multicast groups, and the dynamic update of the virtual structures produced by clustering algorithms. We also need to incorporate these virtual structures into existing reliable multicast protocols to test their effects on the protocols.

References

[1] Matula, D. W., *Cluster Analysis via Graph Theoretic Techniques,* In Proceedings of Louisiana Conference on Combinatorics, Graph Theory and Computing, K. B. Reid, and D. P. Roselle, Editors, University of Manitoba, Winnipeg, 1970, pp.199–212.

[2] Matula, D. W., *k-Components, Clusters and Slicings in Graphs,* SIAM J. Appl. Math., 22(3), 1972, pp.459–480.

[3] Ben-Dor, A., Z. Yakhini, *Clustering gene expression patterns,* In Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB'99, Lyon, France, Aug. 11-14, 1999)

[4] Gibson, D., J. Kleinberg, P. Raghavan, *Two Algorithms for Nearest-Neighbor Search in Hign Dimensions,* In Proceedings of the 29th ACM Symposium on Theory of Computing, 1997, pp. 599-608.

[5] Hartuv, E., A. Schmitt, J. Lang, et al., *An Algorithm for Clustering cDNAs for Gene Expression Analysis,* In Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB'99, Lyon, France, Aug. 11-14, 1999)

[6] Zhang, T., R. Ramakrishnan, M. Livny, *Birch: An Efficient Data Clustering Method for Very Large Databases,* In Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'96, Montreal, Canada, June 2, 1996) pp.103–114.

[7] Krishna, P., N. Vaidya, M. Chatterjee, and D. Pradhan, *A Cluster-based Approach for Routing in Dynamic Networks,* In Computer Communication Review, 27(2), April, 1997. ACM New York, N. Y. pp.49-65.

[8] Levine, B., D. Lavo, and J. J. Garcia-Luna-Aceves, *The Case for Reliable Concurrent Multicasting Using Shared ACK Trees,* In Proceedings of ACM Multimedia (Boston, Massechusettes, USA, Nov. 18-22, 1996). ACM New York, N. Y., pp.365–376.

[9] Lin, J. C. and S. Paul, *RMTP: a Reliable Multicast Transport Protocol,* In Proceedings of the 1996 INFOCOM Conference on Computer Communications (San Francisco, CA, Mar. 24-28, 1996). IEEE Piscataway, N. J., pp.1414–1424.

[10] Motyčková, L. and D. Carr. 2000. *Locality Issues in Reliable Multicasting,* In Proceedings of the 6th Asia-Pacific Conference on Communications (APCC2000, Seoul, South Korea, Oct. 30 - Nov. 2, 2000), pp.800–804.

[11] Motyčková, L., D. Carr, and E. Jennings, *Cluster-Ring Topology for Reliable Multicasting,* In Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (Las Vegas, Nevada, USA. June 26-29, 2000). pp.999-1006

[12] Calvert, K., M. Doar and E. W. Zegura, *Modeling Internet Topology,* IEEE Communications Magazine, June, 1997, pp.160–163.